Write a lexical scanner in C++ to scan a BASIC program, separating the source code into tokens, classifying the tokens into their appropriate token types. For this task, read in a state table (state.dat) which will help you to classify the tokens as to their token type (use enumerated types!): newtoken, resword, variable, integer, real, or delimiter. You will also use an action table (action.dat) which will signify the action to take place as each character is read. These tables (see next page) will be covered in more detail in class. The reserved words for the language will also be read in from their own data file (reserve.dat):

```
print
input
end
read
goto
for
if
then
step
rem
to
next
or
and
```

The source code file (prog1.bas) you will scan:

```
10 rem
11 input x1, y
12 if x1 < y then goto 15
13 zee = x1 - y
14 goto 16
15 zee = x1 + 2 \times y
16 print x1, y, zee
17 for i = 1 to 16 step 5
18 print i
19 next i
20 input st%
21 print "$", st%
22 input x
23 if y >= x1 then x = y^3
24 print x1, #
25 end
```

Your output will be: the list of reserved words for the language in alphabetic order, the state table with appropriate column and row headers, the action table (with appropriate titles, etc) with the explanations of the actions, followed by each token with its appropriate token type.
## State Table

### Character Classification

<table>
<thead>
<tr>
<th>Letter</th>
<th>Digit</th>
<th>Period</th>
<th>Delimiter</th>
<th>Blank</th>
<th>% or $</th>
<th>EOLN</th>
<th>Illegal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Word</td>
<td>Integer</td>
<td>Token</td>
<td>Delimiter</td>
<td>Token</td>
<td>Token</td>
<td>Token</td>
</tr>
<tr>
<td>Reserved Word</td>
<td>Reserved</td>
<td>New</td>
<td>Token</td>
<td>Delimiter</td>
<td>New</td>
<td>New</td>
<td>New</td>
</tr>
<tr>
<td></td>
<td>Word</td>
<td>Variable</td>
<td>Token</td>
<td>Delimiter</td>
<td>Token</td>
<td>Token</td>
<td>Token</td>
</tr>
<tr>
<td>Variable</td>
<td>Variable</td>
<td>Variable</td>
<td>Token</td>
<td>Delimiter</td>
<td>Token</td>
<td>Token</td>
<td>Token</td>
</tr>
<tr>
<td>Integer</td>
<td>Reserved</td>
<td>New</td>
<td>Token</td>
<td>Delimiter</td>
<td>New</td>
<td>New</td>
<td>New</td>
</tr>
<tr>
<td></td>
<td>Word</td>
<td>Integer</td>
<td>Token</td>
<td>Delimiter</td>
<td>Token</td>
<td>Token</td>
<td>Token</td>
</tr>
<tr>
<td>Real</td>
<td>Reserved</td>
<td>New</td>
<td>Token</td>
<td>Delimiter</td>
<td>Token</td>
<td>Token</td>
<td>Token</td>
</tr>
<tr>
<td></td>
<td>Word</td>
<td>Real</td>
<td>Token</td>
<td>Delimiter</td>
<td>Token</td>
<td>Token</td>
<td>Token</td>
</tr>
</tbody>
</table>

## Action Table

### Character Classification

<table>
<thead>
<tr>
<th>Letter</th>
<th>Digit</th>
<th>Period</th>
<th>Delimiter</th>
<th>Blank</th>
<th>% or $</th>
<th>EOLN</th>
<th>Illegal</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Token</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Reserved Word</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>Variable</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>3</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>Integer</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Real</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Delimiter</td>
<td>9</td>
<td>9</td>
<td>4</td>
<td>12</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

## Explanations of Action Table Entries

1. Load character and continue
2. Compare with table of reserved words, print token, reinitialize token
3. Print token, reinitialize token
4. Print token, write "Improper usage", print character, reinitialize token
5. Write "Improper usage", print character
6. Continue
7. Write "Illegal character", print character
8. Compare with table of reserved words, print token, reinitialize token, load char
9. Print token, reinitialize token, load character
10. Load character, change state to variable, print token, reinitialize token
11. Compare with table of reserved words, print token, write "Illegal character", print char
12. Load character, print token, reinitialize token
13. Print Token, write "Illegal Character", print character, reinitialize token